High-Fidelity Radiosity Rendering at Interactive Rates

Stephen Hardt Seth Teller
MIT Synthetic Imagery Group

Abstract

Existing radiosity rendering algorithms achieve interactivity or high fidelity, but not both. Most radiosity renderers optimize interactivity by converting to a polygonal representation and Gouraud interpolating shading samples, thus sacrificing visual fidelity. A few renderers achieve improved fidelity by performing a per-pixel irradiance "gather" operation, much as in ray-tracing. This approach does not achieve interactive frame rates on existing hardware.

This paper bridges the gap, by describing a data structure and algorithm which enable interactive, high-fidelity rendering of radiosity solutions. Our algorithm "factors" the radiosity rendering computation into two components: an *offline* phase, in which a per-surface representation of irradiance is constructed; and an *online* phase, in which this representation is rapidly queried, in parallel, to produce a radiosity value at each pixel. The key components of the offline phase are a heuristic discontinuity ranking algorithm, which identifies the strongest discontinuities, and a hybrid quadtree-mesh data structure which prevents combinatorial interactions between most discontinuities. The online phase involves a novel use of perspective-correct texture-mapping hardware to produce nonlinear, analytic shading effects.

1 Introduction

1.1 Some Limitations of Polygon Hardware

Modern graphics workstations, although extraordinarily powerful, can be ill-suited for viewing global illumination solutions. Polygon hardware typically linearizes both geometry and shading, whereas the illumination function over a surface will in general have discontinuities along curved contours, and vary in a non-polynomial (and certainly non-linear) fashion even where it is smooth. Moreover, screen-space interpolation is not invariant under general viewing transformations, causing shading artifacts during interactive viewing.

Polygon-rendering hardware has been successfully used in interactive walkthroughs of globally illuminated environments [1, 5, 10]. In these interactively rendered sequences, however, the surface geometry is a collection of polygons, and the surface shading is a screen-space linear interpolation of a function whose value is specified at three points (typically, the vertices of a triangle) [2]. Although higher-order geometry primitives exist on some architectures [21], even these polygonalize, then Gouraud interpolate over, the interiors of the resulting triangles. Graphics hardware architectures that perform higher-order shading have been built [9, 17], but are not widely available.

These facts partially explain why many of the beautiful images published in the global illumination literature (e.g., [19]) are produced by ray-casting algorithms which, at each pixel, identify surface points to be shaded, then compute analytical irradiance values there with an object-space algorithm. Given the solution data, this rendering process can consume tens of seconds or minutes per image, depending on scene complexity.

1.2 Towards Accurate Interactive Display

We discuss the generation of accurate irradiance and radiosity values for a polyhedral scene rendered from an interactively-controlled viewpoint. Our implementation uses standard rendering hardware as a massively parallel geometric query engine operating upon a large, object-space, spatial data structure.

We assume that a hierarchical radiosity solution method is in use, which produces as output a discretization of the input into *elements* annotated by radiosity values, and an organization of interactions between elements into *links* annotated by *blockers* (as in [14, 26]). From interactions among the blockers and light sources, we rank irradiance discontinuities by their strength on the receiver, and select the strongest. The selected discontinuities partition each solution element into disjoint regions, inside each of which none of the (selected) discontinuities can be present. We then approximate the irradiance function over each region using a polynomial *interpolant*, whose domain is the surface itself.

In an interactive session, a synthetic eyepoint moves through the scene under user control. In a novel use of rendering hardware (normally used to display perspective-correct textured polygons), the radiance data structure is queried in object space, at every pixel. Next, a host-parallel pass through the query structure generates the radiosity at each pixel from the relevant irradiance interpolant and the surface's reflectance and emittance. Our prototype implementation simultaneously captures global lighting effects and evaluates superlinear radiosity interpolants at interactive rates.

1.3 Algorithmic Foundations

Our algorithms and implementation build upon several ideas and techniques from the literature:

Hierarchical radiosity. We adopt the algorithms of [14, 11, 12, 26] and use them as a hierarchical radiosity solver. We assume that these algorithms converge to an accurate radiosity solution, but do not consider convergence or solver accuracy issues here.

Irradiance data structure. We use the idea of a per-surface data structure which approximates spatially varying irradiance [27, 19, 20, 3].

Discontinuity identification. Our algorithms explicitly identify irradiance discontinuities in order to improve the visual fidelity of the computed solution, as have [4, 15, 22, 20, 25, 8].

Hardware acceleration of object-space computations. As did the "hemi-cube" [7], "two-pass" [23], and "3D painting" [13] algorithms, we use fast graphics hardware to discretize and accelerate object space computations.

Backprojection of occluders. We use the notion of "backprojection" for the computation of accurate source-to-point irradiance in the presence of occluders [8].

This paper introduces several new ideas and techniques, among them:

Discontinuity ordering. We select from a collection of irradiance discontinuities via a heuristic estimate of their relative strengths at the receiver. Although our irradiance gradient is heuristic, it is less computationally intricate than those of [16, 27, 3]. Moreover, both discontinuities caused by emitters and reemitters are handled.

Hybrid mesh structures. Quadtrees are fundamentally unable to model general domains, except by a sort of generalized (and aliasing-prone) binary subdivision. A hybrid of quadtree and explicit meshing yields a meshing scheme more flexible than quadtrees, yet more efficient than explicit meshing.

Hardware acceleration of irradiance queries. We describe a novel use of the polygon-rendering and texture-mapping capabilities of a high-end graphics workstation to generate real-time irradiance queries, in parallel, for all pixels of an image. Our approach avoids both direct rendering of polygonalized elements, and the use of screen-space interpolation (i.e., Gouraud shading).

2 High Fidelity Rendering

This section describes our interpolant data structure, construction scheme, and rendering algorithm. The construction scheme requires:

1. a set of solution elements from an HR algorithm;
2. an algorithm for identifying discontinuities, and sorting them in order of strength;
3. a function `IrradianceAtPoint()` which computes the irradiance at any source point due to all receivers irradiating that point.

We first show how to generate a data structure which accurately represents irradiance, and how to use this structure for rapid rendering. Methods for generating (1), (2), and (3) then follow in Sections 3, 4, and 5 respectively.

2.1 Data Structure

The structure is a list of triangles, each annotated with a quadratic *interpolant* for irradiance (Figure 1); an expression in s and t which smoothly approximates irradiance over the domain region. We use quadratic interpolants, as we have found that constant and linear interpolants are inadequate to capture the radiosity function faithfully, even in regions where it varies smoothly, and that higher order interpolants do not significantly improve the interpolant fit.

For each (quadtree) solution element, we *rank* the discontinuities affecting the element and select those with the strongest effect on the receiver. These discontinuity segments form the input to a constrained Delaunay triangulation algorithm [19], which produces a triangulation containing the segments, along with adjacency information about the triangles. These triangles are then assembled into a list, and an interpolant constructed for each triangle (Figure 2).

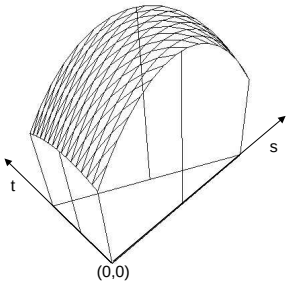


Figure 1: Quadratic interpolant. The six uprights represent the sample values.

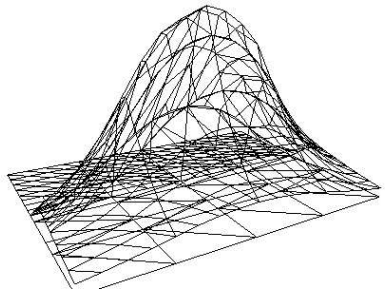


Figure 2: All interpolants on a quadtree.

2.2 Constructing Interpolants

For each triangle, `IrradianceAtPoint()` is invoked at the triangle vertices and edge midpoints to collect six irradiance values $\mathbf{r_i}$. Using these six values, we construct an irradiance interpolant over the entire triangle, as a function of barycentric coordinates (s, t).

Given six barycentric sample locations $p_i = (s_i, t_i)$ and corresponding values r_i, the interpolant construction must determine coefficients $A...F$ of the function

$$R(s,t) = As^2 + 2Bst + 2Cs + Dt^2 + 2Et + F \tag{1}$$

so that

$$R(s_i, t_i) = r_i, \qquad 0 \le i < 6.$$

In general, this requires the solution of the quadratic form

$$\begin{pmatrix} s_i & t_i & 1 \end{pmatrix} \begin{pmatrix} A & B & C \\ B & D & E \\ C & E & F \end{pmatrix} \begin{pmatrix} s_i \\ t_i \\ 1 \end{pmatrix} = r_i, \qquad 0 \le i < 6.$$

However, judicious choice of a barycentric coordinate system and sample locations (the triangle vertices and edge midpoints) reduces the problem to solving a system of six linear equations. We write

$$\begin{pmatrix} R(0,0) \\ R(\frac{1}{2},0) \\ R(1,0) \\ R(\frac{1}{2},\frac{1}{2}) \\ R(0,1) \\ R(0,\frac{1}{2}) \end{pmatrix} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 \\ \frac{1}{4} & 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 2 & 0 & 0 & 1 \\ \frac{1}{4} & \frac{1}{2} & 1 & \frac{1}{4} & 1 & 1 \\ 0 & 0 & 0 & 1 & 2 & 1 \\ 0 & 0 & 0 & \frac{1}{4} & 1 & 1 \end{pmatrix} \begin{pmatrix} A \\ B \\ C \\ D \\ E \\ F \end{pmatrix} = \begin{pmatrix} r_0 \\ r_1 \\ r_2 \\ r_3 \\ r_4 \\ r_5 \end{pmatrix}$$

Inverting the 6×6 matrix symbolically and multiplying by the sample vector yields the closed form solution for the coefficients:

$$\begin{pmatrix} A \\ B \\ C \\ D \\ E \\ F \end{pmatrix} = \begin{pmatrix} 2r_0 - 4r_1 + 2r_2 \\ 2r_0 - 2r_1 + 2r_3 - 2r_5 \\ -\frac{3}{2}r_0 + 2r_1 - \frac{1}{2}r_2 \\ 2r_0 + 2r_4 - 4r_5 \\ -\frac{3}{2}r_0 - \frac{1}{2}r_4 + r_5 \\ r_0 \end{pmatrix}$$

The six resulting floating point numbers $\begin{pmatrix} A & 2B & 2C & D & 2E & F \end{pmatrix}$ are then stored as the interpolant for the triangle in question. (Storing $2B$, $2C$, and $2E$, rather than B, C, and E, avoids three multiplies during subsequent evaluation.)

2.3 Rendering

We now have a list L of triangles, each annotated with an irradiance interpolant expressed in terms of triangle barycentric coordinates. Our goal is to assign each pixel P in the output image the radiosity value for that point B (on the visible triangle T) that projects to P. We do so with a multi-pass algorithm on a graphics workstation.

To generate each frame of an interactively rendered sequence, we:

(a) Generate an image $\mathcal{I}_1$ in which the value at each pixel P encodes which triangle T is visible at P;

(b) Generate an image $\mathcal{I}_2$ in which the value at each pixel P encodes the barycentric coordinates B of the object-space point that projects to P;

(c) Using $\mathcal{I}_1$ and $\mathcal{I}_2$, generate the output image $\mathcal{O}$ by looping over all pixels P and evaluating T's irradiance interpolant at the object-space point corresponding to P;

(d) Copy $\mathcal{O}$ to the framebuffer, and swap it forward for display.

2.4 Visible Triangle Determination

Task **(a)**, visible triangle determination, is accomplished with the polygon fill and depth-buffering hardware. Each triangle T is rendered as a solid "color," a 24-bit encoding of the index of T in the list L (Color Plate A). This rendering is done in the hardware backbuffer to avoid overwriting the frame currently displayed in the frontbuffer. Occlusion is handled properly by the depth-buffering hardware. After all triangles have been rendered, the framebuffer is copied to host memory to create $\mathcal{I}_1$.

2.5 Barycentric Coordinate Determination

To accomplish task **(b)**, we use a $256 \times 256 \times 32$ bit texture map. This texture consists simply of an encoding of s and t at texel (s,t). We render each element triangle, issuing it with (floating-point) texture coordinates $(0,0)$, $(1,0)$, and $(0,1)$. The perspective-correct texture-mapping hardware deposits, at each pixel, the barycentric coordinates of the point on the visible triangle (Color Plate B). Note that Gouraud-interpolation hardware cannot be used for object-space interpolation, since it interpolates in screen space. Again, rendering is done in the backbuffer and occlusion is handled properly by the depth-buffering hardware. Finally, after all triangles have been rendered, the framebuffer is copied to host memory to create $\mathcal{I}_2$.

2.6 Evaluating the Interpolant and Radiosity

To complete task **(c)**, we loop over every pixel P in the output image $\mathcal{O}$. The corresponding pixels in the scratch images $\mathcal{I}_1$ and $\mathcal{I}_2$ give the visible triangle T (and its interpolant) at P and the barycentric coordinates B on T of the point that projects to P, respectively. T's interpolant is evaluated at B to produce irradiance, which is then multiplied by reflectivity and summed with emissivity to produce radiosity. Given (s,t), evaluating

$$R(s,t) = As^2 + 2Bst + 2Cs + Dt^2 + 2Et + F$$

requires only eight multiplies and five adds (the factors of two are folded into the stored coefficients).

2.7 Depositing the Rendered Pixels

The final step, part **(d)**, simply copies $\mathcal{O}$ to the display backbuffer and swaps front and back buffers for an instantaneous update (Color Plate C).

2.8 Costs and Parallelism

Steps **(a)** and **(b)** consist of flat-shaded polygon rendering, texture mapped polygon rendering, and copying pixels from the framebuffer to host memory. Step **(d)** consists of copying pixels from host memory to the framebuffer. These operations are all extremely fast (i.e. can easily be performed at interactive rates) on a high-end graphics workstation such as the Reality Engine [2].

The bottleneck of our method is step **(c)**, taking $10-100$ times as long as the other steps. Fortunately, this operation is highly parallelizable. The color of each pixel depends only on the triangle list and the scratch images $\mathcal{I}_1$ and $\mathcal{I}_2$. Since the color of every pixel in the output image is independent of the color of every other pixel, there is no data dependency problem. The time to evaluate the radiosity for any pixel is constant, so there is no load balancing problem.

On our host-parallel system, we create a separate evaluation process for each of the N physical processors and partition all pixels on the screen equally among them. Our implementation uses four processors (RISC R4400s running at 250 MHz), shared memory for communication between processes, and hardware locks for synchronization.

3 Radiosity Solution

The underlying radiosity solver is a reimplementation of the hierarchical radiosity and wavelet radiosity algorithms described in [14, 11, 26]. Solving the global pass of the radiosity solution provides us with

- A quadtree subdivision of the input geometry;

- A radiosity value for each quadtree node;
- A "link" representing each element-element interaction;
- A set of "blockers" for each link.

This computed solution satisfies requirement (1) for the high fidelity rendering algorithm; that is, it is a suitable input for interpolant construction.

4 Selecting Discontinuities

4.1 Motivation

The irradiance function has discontinuities due to contact and occlusion. Since smooth interpolants do not perform well in the presence of discontinuities, researchers have proposed the construction of "discontinuity meshes," in which the solution elements (i.e., function domains) are explicitly meshed, in order to introduce boundary curves wherever discontinuities are detected [4, 19, 15, 20, 22, 25, 3, 8]. Once discontinuities have been banished from the interior of the element, a smooth interpolant can be fit, although for non-trivial domains this requires some fairly complex geometric and topological infrastructure [22, 19, 18].

We assume, as in [14], that from every quadtree solution element all sources of irradiance there may be found, and that relevant blockers are associated with all source-receiver links. For each receiver element and its links, we identify the curves of irradiance discontinuity on the element. This is done by considering all edge-edge (EE) and vertex-edge (VE) pairs drawn from among the light source and blockers, as in [15, 20]. Currently, we ignore triple-edge (EEE) critical surfaces as these generally have a weak visual effect.

A general quadtree element, attempting to capture irradiance due to a multi-sided light source shining past some number of blockers, may intersect many discontinuity surfaces. However, quadtree subdivision of a node will tend to reduce the number of discontinuities impinging on the node's children.

4.2 Discontinuity Ranking

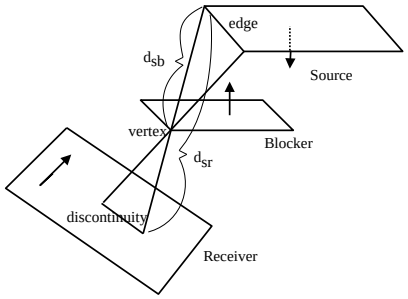


Figure 3: An EV(edge-vertex) discontinuity.

Geometric interactions (horizons and occlusion) tend to produce an enormous number of discontinuity surfaces in a typical scene, and many of these surfaces will intersect a typical receiver surface. However, most of the geometric discontinuities will be quite weak radiometrically; consequently, they will have little or no visually discernible effect. We propose a method for *ranking* discontinuities by a heuristic "weight," w, defined as follows:

$$w = \max\left(\frac{d_{sb}}{d_{br}}\right) \cdot B_{src}$$

Above, B_{src} is the radiosity of the source, d_{sb} and d_{sr} are the distances along a line in the VE or EV swath from the source to blocker and from the source to receiver, respectively, and maximization occurs over all pairs of source-blocker vertices involved in the discontinuity. Figure 3 shows the configuration for an EV discontinuity. VE discontinuities are handled similarly.

This weight is the product of a geometric and a radiometric factor. The geometric factor, $\max\left(\frac{d_{sb}}{d_{br}}\right)$, is proportional to the rate at which the source becomes visible (or obscured) as seen by an observer crossing the discontinuity. The radiometric factor is B_{src}; the brighter the source, the stronger the discontinuity.

For each leaf in the quadtree, we compute the k worst discontinuities of weight at least w_{min}, where k and w_{min} are parameters to the algorithm (Figures 4, 5, and 6). (The minimum weight criteria avoids expending computational effort meshing solution elements which are impinged upon only by weak discontinuities.)

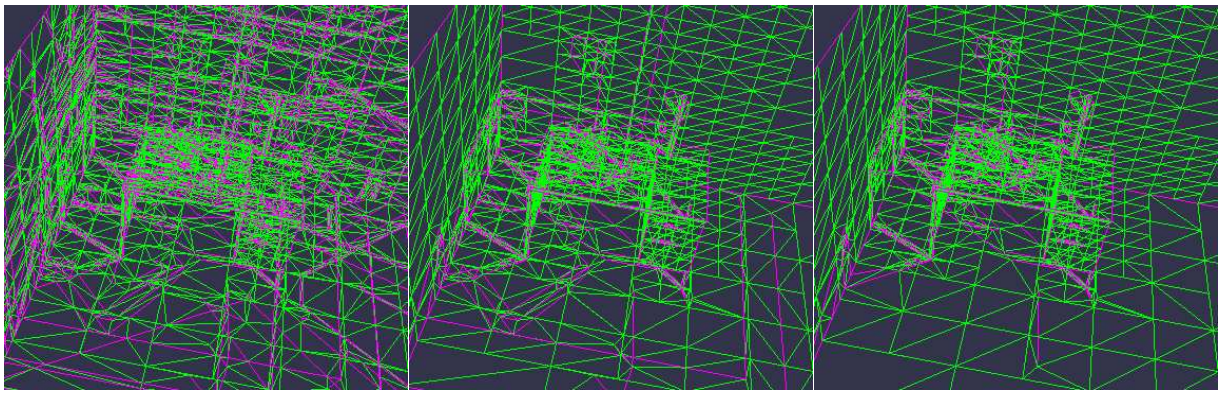


Figure 4: $k = 10$, $w_{min} = 0$.

Figure 5: $k = 10$, $w_{min} = 10$.

Figure 6: $k = 10$, $w_{min} = 100$,

This identifying and ranking scheme for discontinuities impinging on each quadtree leaf satisfies requirement (2) of the high fidelity rendering algorithm.

5 Accurate Point Irradiance

In the spirit of "two-pass" methods [23, 19], we use the coarse hierarchical radiosity solution to compute more accurate radiosity values at specific points on the geometry. For a point x, we compute the point-to-polygon form factors from x to all sources visible from x [6]. To achieve point-to-source form factors with accurate visibility, we "backproject" potential blockers to the source [8], and add only irradiance from unobscured fragments.

This irradiance gathering operation satisfies requirement (3) of the high fidelity rendering algorithm.

6 Implementation

We implemented these algorithms on a Silicon Graphics Reality Engine with four 250 MHz MIPS RISC R4400 CPUs and 512Mb of memory. The underlying radiosity solver is a reimplementation, in C++, of the hierarchical radiosity and wavelet radiosity algorithms described in [14, 11, 26]. The system components and code complexity are as follows:

- Form factor and radiosity solver (18,000 lines of C++);
- Interpolant module (1500 lines of C++);

- User interface (4500 lines of C++);
- Rendering module (3000 lines of C++);
- Computational geometry and math modules (3000 lines of C++).

Our test scene, comprised of about sixty quadrilaterals, is shown in Color Plate D. The hierarchical radiosity algorithm, with the allowable error set to $0.1\mathrm{W/M}^2$, the maximum number of discontinuities per receiver set to 10, and the minimum discontinuity weight set to 100, ran to convergence on this input in less than two minutes, and meshed the input surfaces into 1622 quadtree (leaf) elements. After triangulation, there were 6576 triangles (interpolants), an average of about 4 triangles per element. Figure 6 shows the resulting quadtrees and triangulations. Interpolant construction required two and a half hours of CPU time (running on a single processor), about eighty times the cost of computing the radiosity solution.

Color Plate D is a screen snapshot taken from an interactive session viewing the office model at NTSC (640×480) resolution. Our real-time rendering algorithm achieved 2.3 updates per second for this model. This simple office scene serves to highlight the discontinuity resolution and shading abilities of the techniques described here. Note that the mesh (Figure 6) is relatively coarse, but still yields a crisp image due to the use of irradiance interpolants.

Figures 7 and 8 show a detail view of the corner near the light source, which contains a strong horizon discontinuity. The difference between Gouraud-shaded rendering and interpolant rendering is particularly evident in this region, and on the walls near the desk and desk lamp.

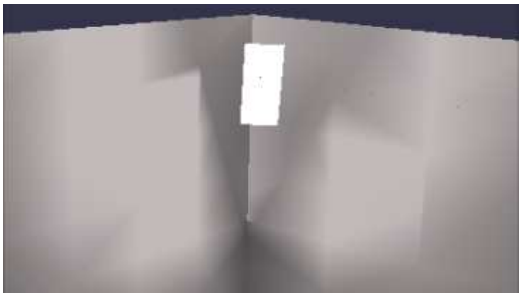


Figure 7: Gouraud shading.

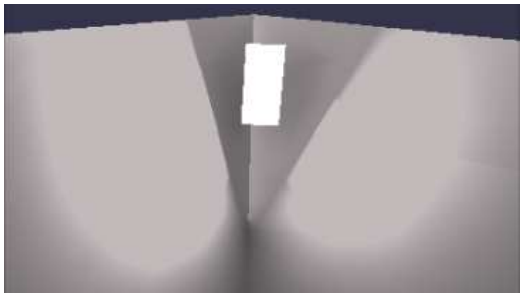


Figure 8: Quadratic interpolant rendering.

Our implementation has at least two limitations, namely 1) it processes only polyhedral scenes, and 2) since our techniques rely on graphics hardware, they operate with relatively low numerical precision. However, we expect the latter concern to be ameliorated by the increasing precision of next-generation software and hardware architectures. Generalizing to non-polyhedral geometries remains a difficult problem.

7 Future Work

7.1 Improved Weighting Metric

We are experimenting with an improved method for ranking discontinuities by their worst-case radiometric "weight." The weight of a discontinuity produced by a specific source, blocker, receiver combination is derived by bounding the maximum change in the radiosity function, per unit length on the receiver. This weight will be proportional to the source radiosity, and to the change in solid angle subtended by the source as viewed from the receiver, as a result of motion on the receiver across the discontinuity.

7.2 Transport vs. Representation Error

The transport-based error criteria used by hierarchical radiosity algorithms is extremely conservative, and can do far more work on a surface than is required to capture the irradiance there faithfully (consider hundreds of strong sources arranged so as to produce nearly constant irradiance; a transport-based HR algorithm would subdivide to great depth to perform each source–receiver transport accurately). Clustering techniques such as [24] ameliorate this disadvantage somewhat, but still do not drive subdivision based on representation error, a quantity which global illumination algorithms strive to minimize.

We believe that a subdivision criteria based on representational error – that is, an estimate of the error between represented radiosity and that due to all sources irradiating the receiver – will be necessary to achieve algorithms that are efficient for a broad range of scenes. We have implemented a strategy for representation-based subdivision, with promising initial results for small scenes.

7.3 Extension to Radiance

Radiosity algorithms are giving way to those for radiance, a much more complex 4-dimensional quantity associated with each surface point's position, and the direction from which the point is viewed. We plan to extend our algorithms to operate on a radiance data structure, by rendering (a discretized representation of) the direction (θ, ϕ) from which each surface point is viewed, in object space. The assembled values (s, t, θ, ϕ) will then be used to evaluate a suitable 4-dimensional quadratic radiance interpolant.

8 Conclusion

Generating high-quality imagery that precisely captures diffuse irradiance is a computationally expensive proposition. We presented a data structure which captures a representation of irradiance for every point on every surface in a scene. Later, in combination with a fast massively parallel graphics hardware rendering architecture, the data structure is queried, in parallel, to produce quadratically interpolated radiosity values at interactive rates.

One of the scheme's strengths, its use of rendering hardware, can also be considered a limitation due to that hardware's limited precision. However, software advances (e.g., OpenGL) and hardware augmentations (e.g., higher iterator precision and framebuffer resolution, larger textures, and object-space "Gouraud" interpolation) should make these techniques both more efficient and accurate.

The realization of this technique required advances at both theoretical and practical levels. The theoretical advances of this paper were the ranking of discontinuities by relative strengths, and a "factoring" of radiosity rendering into online and offline components. The practical advances were the use of texture-mapping hardware for barycentric coordinate generation, and the introduction of a hybrid quadtree-mesh, a quadtree with discontinuity-meshed leaves,

References

[1] Airey, J. M., Rohlf, J. H., and Brooks, Jr., F. P. Towards Image Realism with Interactive Update Rates in Complex Virtual Building Environments. *ACM Siggraph Special Issue on 1990 Symposium on Interactive 3D Graphics 24*, 2 (1990), 41–50.

[2] Akeley, K. RealityEngine Graphics. *Computer Graphics (Proc. Siggraph '93)* (1993), 109–116.

[3] Arvo, J. The Irradiance Jacobian for Partially Occluded Polyhedral Sources. In *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)* (July 1994), A. Glassner, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, ACM Press, pp. 343–350. ISBN 0-89791-667-0.

[4] Baum, D., Mann, S., Smith, K., and Winget, J. Making Radiosity Usable: Automatic Preprocessing and Meshing Techniques for the Generation of Accurate Radiosity Solutions. *Computer Graphics (Proc. Siggraph '91) 25*, 4 (1991), 51–60.

[5] Baum, D., and Winget, J. Real Time Radiosity Through Parallel Processing and Hardware Acceleration. *Computer Graphics (1990 Symposium on Interactive 3D Graphics) 24*, 2 (March 1990), 67–75.

[6] Baum, D. R., Rushmeier, H. E., and Winget, J. M. Improving Radiosity Solutions Through the Use of Analytically Determined Form Factors. *Computer Graphics (Proc. Siggraph '89) 23*, 3 (1989), 325–334.

[7] Cohen, M., and Greenberg, D. The Hemi-Cube: A Radiosity Solution for Complex Environments. *Computer Graphics (Proc. Siggraph '85) 19*, 3 (1985), 31–40.

[8] Drettakis, G., and Fiume, E. A Fast Shadow Algorithm for Area Light Sources Using Backprojection. In *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)* (July 1994), A. Glassner, Ed., Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, ACM Press, pp. 223–230. ISBN 0-89791-667-0.

[9] Fuchs, H., Poulton, J., Eyles, J., Greer, T., GoldFeather, J., Ellsworth, D., Molnar, S., Turk, G., Tebbs, B., and Israel, L. Pixel-Planes 5: A Heterogeneous Multiprocessor Graphics System Using Processor-Enhanced Memories. *Computer Graphics (Proc. Siggraph '89) 23*, 3 (1989), 79–88.

[10] Garlick, B., Baum, D. R., and Winget, J. M. Interactive Viewing of Large Geometric Databases Using Multiprocessor Graphics Workstations. *Siggraph '90 Course Notes (Parallel Algorithms and Architectures for 3D Image Generation)* (1990).

[11] Gortler, S., Schröder, P., Cohen, M., and Hanrahan, P. Wavelet radiosity. *Computer Graphics (Proc. Siggraph '93)* (August 1993), 221–230.

[12] Haines, E., and Wallace, J. Shaft Culling for Efficient Ray-Traced Radiosity. In *Proc. 2^{nd} Eurographics Workshop on Rendering* (May 1991).

[13] Hanrahan, P., and Haeberli, P. E. Direct WYSIWYG Painting and Texturing on 3D Shapes. In *Computer Graphics (SIGGRAPH '90 Proceedings)* (Aug. 1990), F. Baskett, Ed., vol. 24, pp. 215–223.

[14] Hanrahan, P., Salzman, D., and Aupperle, L. A Rapid Hierarchical Radiosity Algorithm. *Computer Graphics (Proc. Siggraph '91) 25*, 4 (1991), 197–206.

[15] Heckbert, P. Discontinuity Meshing for Radiosity. *Third Eurographics Workshop on Rendering* (May 1992), 203–226.

[16] Holzschuch, N., and Sillion, F. Accurate Computation of the Radiosity Gradient for Constant and Linear Emitters. In *Rendering Techniques '95 (Proceedings of the Sixth Eurographics Workshop on Rendering)* (New York, 1995), P. M. Hanrahan and W. Purgathofer, Eds., Springer-Verlag, pp. 186–195.

[17] Kirk, D., and Voorhies, D. The rendering architecture of the DN10000VS. *Computer Graphics (Proc. Siggraph '90) 24*, 4 (1990), 299–307.

[18] Lischinski, D. Incremental Delaunay Triangulation. In *Graphics Gems IV*, P. Heckbert, Ed. AP Professional, 1994, pp. 47–59.

[19] Lischinski, D., Tampieri, F., and Greenberg, D. P. Discontinuity Meshing for Accurate Radiosity. *IEEE Computer Graphics and Applications 12*, 6 (1992), 25–39.

[20] Lischinski, D., Tampieri, F., and Greenberg, D. P. Combining Hierarchical Radiosity and Discontinuity Meshing. *Computer Graphics (Proc. Siggraph '93) 27* (1993).

[21] Neider, J., Davis, T., and Woo, M. *OpenGL Programming Guide.* Addison-Wesley, 1993.

[22] Salesin, D., Lischinski, D., and DeRose, T. Reconstructing Illumination Functions with Selected Discontinuities. In *Proc. 3^{rd} Eurographics Workshop on Rendering* (May 1992), pp. 99–112.

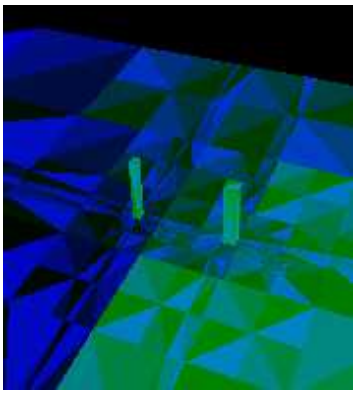
[23] Sillion, F. X., and Puech, C. A General Two-Pass Method Integrating Specular and Diffuse Reflection. In *Computer Graphics (SIGGRAPH '89 Proceedings)* (July 1989), J. Lane, Ed., vol. 23, pp. 335–344.

[24] Smits, B., Arvo, J., and Greenberg, D. A Clustering Algorithm for Radiosity in Complex Environments. *Computer Graphics (Proc. Siggraph '94) 28* (1994).

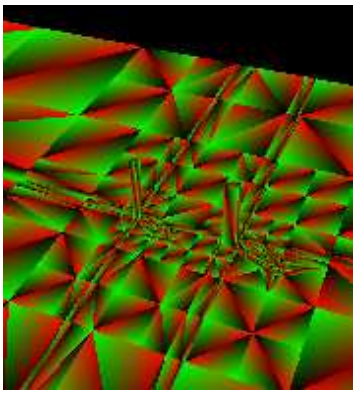
[25] Teller, S. Computing the Antipenumbra Cast by an Area Light Source. *Computer Graphics (Proc. Siggraph '92) 26*, 2 (1992), 139–148.

[26] Teller, S., and Hanrahan, P. Global Visibility Algorithms for Illumination Computations. *Computer Graphics (Proc. Siggraph '93) 27* (1993), 239–246.

[27] Ward, G. J., and Heckbert, P. Irradiance Gradients. *Third Eurographics Workshop on Rendering* (May 1992), 85–98.



Triangle identifiers
(Hardt and Teller,
Color Plate A).



Barycentric coordinates
(Hardt and Teller,
Color Plate B).

Quadratic interpolant rendering
(Hardt and Teller,
Color Plate C).

One frame of an interactive viewing session.
(Hardt and Teller, Color Plate D).